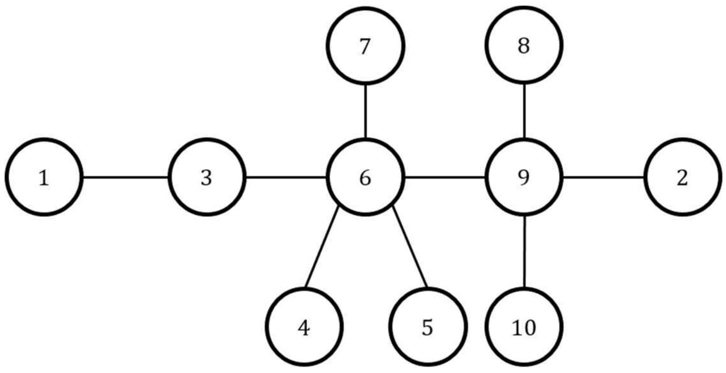
Problem Set

Please check that you have 11 problems that are spanned across 27 pages in total (including Korean translation and this cover page).

Problem A
Gourmet Tour
Time Limit: 0.5 Seconds

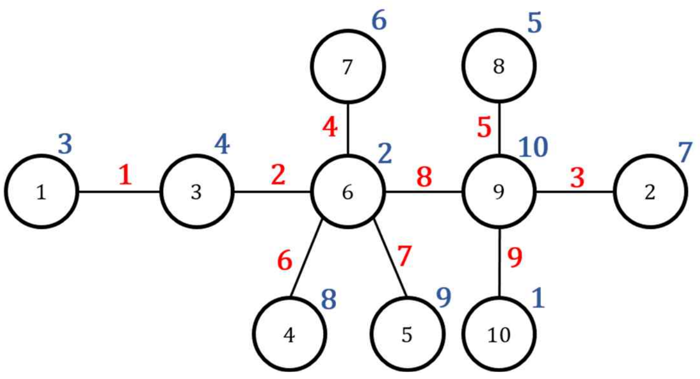
Minsu, who is attending university in Seoul, is planning a gourmet trip to Busan by train for the summer vacation. After researching all the restaurants along the way from Seoul to Busan, he finds that most of them are located in cities with stations on the Gyeongbu(GB) train line, and there are occasionally restaurants in cities a bit far from the GB train line stations. Minsu decides to visit the restaurants by getting off at the cities with stations while traveling from Seoul to Busan on the GB train line. In the case that a restaurant is in a city a bit far from a station on the GB train line, he plans to get off at the station, takes a taxi to the restaurant, visits it, and then returns to the station to catch the train again. Note that each city has one restaurant where he wants to visit. After successfully completing the trip, Minsu ranks the restaurants he has visited and discovers a curious fact. When he collects the values of the differences in the rankings of the restaurants that are in adjacent cities on his travel route, all the difference values are different. What can have been the rankings of the restaurants that Minsu gives?

Let us represent Minsu's travel route in the form of a graph. The cities with stations on the GB train line or a bit far from that line where he visits restaurants are represented as nodes. Two nodes corresponding to consecutive cities on the GB train line are connected as an edge and two nodes corresponding to a city X on the GB train line and a city Y a bit far from X are also connected as an edge. When the total number of cities where Minsu visits restaurants is n, the graph has n nodes and $n - 1$ edges. Nodes are numbered as distinct integers between 1 and n. For example, the figure below represents Minsu's travel route in the form of an undirected graph with 10 nodes (10 restaurants in 10 cities).



The rankings of restaurants assigned by Minsu are integers from 1 to n without duplication. It can be considered as an assignment of rankings to nodes in the graph. The curious fact that Minsu discovers is that the differences of rankings assigned to any two adjacent nodes are all different. For example, the figure below represents the assigned rankings (blue numbers) and the differences (red numbers) of rankings between adjacent nodes. Note that the differences of rankings are integers from 1 to $n - 1$ without duplication.

Given a travel route graph for n cities, write a program to compute the rankings of n restaurants in the cities satisfying the condition explained above.



Input

Your program is to read from standard input. The first line of input contains the integer n ($3 \leq n \leq 50{,}000$), representing the total number of nodes corresponding to the cities (restaurants) to be visited in the travel route graph. The nodes are numbered as 1 to n. Each of the following $n-1$ lines contains two integers u and v ($1 \leq u \neq v \leq n$), separated by a space, representing an edge connecting two nodes u and v of the travel route graph.

Output

The first line should contain n integers $a_1, a_2, \dots, a_n$, separated by spaces, where a_i represents the ranking of city i and must satisfy the condition mentioned above. Note that a_i is an integer between 1 and n and there is no duplication among a_i's. If there are multiple combinations of rankings that satisfy the condition, output any of them.

The following shows sample input and output for two test cases.

Sample Input 1

```
3
1 2
1 3
```

Output for the Sample Input 1

```
1 3 2
```

Sample Input 2

```
10
1 3
3 6
6 7
6 4
6 5
6 9
9 8
9 10
9 2
```

Output for the Sample Input 2

```
3 7 4 8 9 2 6 5 10 1
```

Problem B
Grid Partition
Time Limit: 2.5 Seconds

Consider a grid of size $n \times n$ which has n^2 cells. Each cell in the grid is labeled with an integer less than or equal to n. Cells with the same label belong to a same group.

If two cells that share a side in the grid belong to the same group, they are said to be *connected*. If cell A is connected to cell B, and cell B and cell C are connected, then cell A is said to be connected to cell C as well.

Given a grid of size $n \times n$, we want to partition the cells in the grid into n groups by assigning labels to cells. The conditions that must be observed during partition are as follows.

1. Each group must contain n cells.
2. Cells belonging to the same group must be connected.

For example, when $n = 3$, the partition shown in Figure B-1(A) does not meet the conditions because cells in the group 1 (cells labeled 1) are not connected. Cells in the group 2 are not connected either. On the other hand, the other partitions (i.e., except (A)) shown in Figure B-1 satisfy the above conditions.

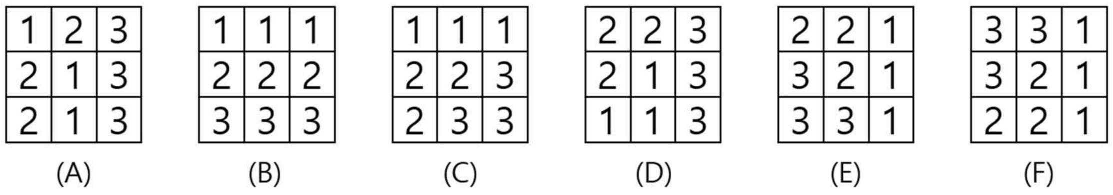


Figure B-1. Examples of grid partition

If you rotate the grid shown in Figure B-1(C) 90 degrees clockwise, it will be equal to Figure B-1(E) and if you flip it upside down, it will be Figure B-1(F). In short, the partition shown in Figure B-1(C) becomes the same partition as the partition shown in Figure B-1(F) through rotation and flipping, so these are considered the *same partition*. In this way, all partitions that obtained by applying consecutive rotations or flips to a partition are considered the same partition.

Comparing the partition shown in Figure B-1(D) with the partition shown in Figure B-1(F), only the assigned group numbers are different, however the partition itself can be seen as the same. Hence, the partition shown in Figure B-1(D) and that in shown B-1(F) are considered the same partition. In this way, grids with different labeling numbers but the same partitioning structure are all the same partition.

To be more precise, partition (C), partition (D), partition (E), and partition (F) shown in Figure B-1 are all the same.

Given an $n \times n$ grid with numbers pre-assigned to some cells, we want to assign a number to each unnumbered cell such that the resulting grid is partitioned so that it satisfies the conditions described above. Partitions can differ depending on how they assign numbers to unnumbered cells.

For example, suppose we are given as input the 4×4 grid shown in Figure B-2. Cells numbered 0 indicate that they are not assigned any number.

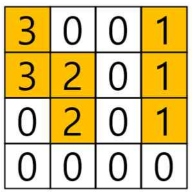


Figure B-2. An example of a 4x4 input grid

Figure B-3 shows three different results for partitioning the grid while maintaining the pre-assigned cell numbers in the input grid shown in Figure B-2.

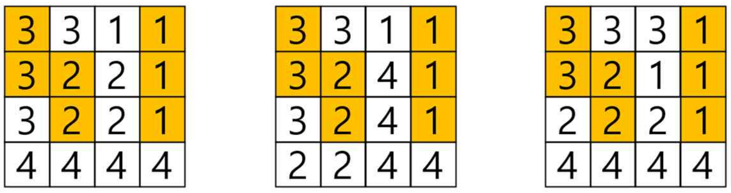


Figure B-3. Different partitions obtained from the input shown in Fig. B-2

Given an integer n representing the size of the grid, a positive integer k, and information on numbers pre-assigned to some cells in the $n \times n$ grid, you are to write a program which computes the total number of different partitions and finds k different partitions while maintaining the pre-assigned cell numbers. If there are no pre-assigned cells in the input grid, your program computes the number of all different partitions that exist in the input grid and finds any k different partitions among all the partitions.

Input

Your program is to read from standard input. The input starts with a line containing two integers, $n(4 \leq n \leq 6)$ and $k(1 \leq k \leq 20)$, where n indicates the size of a grid and k the number of different partitions to print out.

The i-th line of the following n lines contains n integers with no blank each of which is between 0 and n and represents the pre-assigned number for the corresponding cell on the i-th row of the grid. The integer 0 means that the corresponding cell is not assigned any number.

Note that given an input grid with pre-assigned numbers, following properties hold:

1. Cells pre-assigned by the same number are *connected*. In other words, cells in the same group are connected. And cells of number 0, which are actually not pre-assigned any number, are also connected.
2. The number of groups of cells determined by the pre-assigned numbers is 4 at most.

See that the above properties hold for the input shown in Figure B-2.

Output

Your program is to write to standard output. The first line contains the total number of all different partitions maintaining the pre-assigned cell numbers. In the following lines, the k different partitions will be printed.

In the first line of the i-th $(1 \leq i \leq k)$ partition, print '`Partition #i`' as shown in the following samples. In following n lines, print numbers assigned to cells in the grid. Note that the numbers assigned to cells are integers between 1 and n.

Since there may be many different partitions, you may print out any k different partitions in any order. You can assume that k possible partitions always exist for any input.

The following shows sample input and output for one test case.

Sample Input 1

```
4 3
3001
3201
0201
0000
```

Output for the Sample Input 1

```
11
Partition #1
3211
3221
3241
3444
Partition #2
3241
3241
3241
3241
Partition #3
3211
3241
3241
3244
```

Problem C
Happy Point
Time Limit: 1 Second

Jim, a boy in a gloomy mood after the COVID-19 pandemic, has a feeling that all the burden in the world is in his back. He wants to measure the happiness of his friends. One method he devised is to analyze the text messages from his friends to determine if the friends are happy. If a character in the message is included in the word "`HAPPY`," it is a happy character; if a character is in the word "`SAD`," it is a gloomy one. The character '`A`' is included in both words, and it is a happy and gloomy character. The character '`B`' is included in neither words, and it is neither happy nor gloomy.

The degree of happiness of the message is calculated based on the number of happy and gloomy characters of the message. If a character is happy, the happy point P_H is incremented by one; if a character is gloomy, the gloomy point P_G is incremented by one. The degree of happiness H of a message is calculated as follows:

$$H = \frac{P_H}{P_H + P_G}$$

If a character is neither happy nor gloomy, it is not counted as either P_H or P_G. Assume that the degree of happiness is 0.5 if both the happy point and the gloomy point are zero ($P_H = P_G = 0$).

For example, given the message from a friend as follows:

"`SAD MOVIES ALWAYS MAKE ME CRY`"

the happy point of this message can be calculated as 6 since `A` and `Y` appear four and two times, respectively: for `A`, one in `SAD`, two in `ALWAYS`, and one in `MAKE`; for `Y`, one in each of `ALWAYS` and `CRY`. The gloomy point, however, is 8 since `S`, `A`, and `D` appear three, four, and one time, respectively: for `S`, one in each of `SAD`, `MOVIES`, and `ALWAYS`; for `A`, it is calculated as same as that of the happy point; for `D`, one in `SAD`. Since $P_H = 6$ and $P_G = 8$, the degree of happiness is $H = \frac{6}{14} = 0.4286$, that is 42.86%. Therefore, the friend is in a gloomy mood since the degree of happiness is less than 50%.

Given a text message from Jim's friend, write a program to calculate the degree of happiness of the friend based on the message.

Input
Your program is to read from standard input. The input consists of a line containing the message from your friend. The message consists of alphabetic words only. The words are separated by a space. Every word consists of capital letters only. The maximum length of a word is 20 and the maximum number of words is 80.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the degree of happiness calculated. The degree of happiness should be printed in percent without the percent symbol. The degree of happiness should be rounded to two decimal places after the decimal point. Beware that the degree of happiness is 0.5 if both the happy point (P_H) and the gloomy point (P_G) are zero.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
`SAD MOVIES ALWAYS MAKE ME CRY`	`42.86`

Sample Input 2	Output for the Sample Input 2
`ICPC PROGRAMMING`	`75.00`

Sample Input 3	Output for the Sample Input 3
`WRITING`	`50.00`

Problem C

행복 점수

제한 시간: 1 초

코로나 19이후 우울함에 빠져 있는 소년 짐(Jim)은 세상의 모든 짐을 지고 있는 느낌이다. 짐은 친구들의 행복 정도를 측정하고 싶어졌다. 짐이 생각한 한 가지 방법은 친구들이 보낸 문자 메시지를 분석하여 친구들이 행복한지 판단하는 것이다. 문자 메시지의 글자가 단어 "HAPPY"에 나타나면 그 글자는 행복한 글자이고 단어 "SAD"에 나타나면 그 글자는 우울한 글자이다. 글자 'A'는 양쪽에 모두 나타나므로 행복하기도 하고 우울하기도 한 글자이다. 글자 'B'는 어느 쪽에도 나타나지 않으므로 행복하지도 않고 우울하지도 않은 글자이다.

메시지의 행복 지수는 행복 점수와 우울 점수를 이용하여 계산하는데, 행복 점수와 우울 점수는 글자가 행복한 글자인지 우울한 글자인지 여부에 따라 계산한다. 어떤 글자가 행복하면 행복 점수 P_H를 하나 증가시키고 어떤 글자가 우울하면 우울 점수 P_G를 하나 증가시킨다. 행복 지수 P_G는 다음과 같이 계산한다:

$$H = \frac{P_H}{P_H + P_G}$$

행복하지도 않고 우울하지도 않은 글자는 P_H나 P_G 어느 쪽에도 계산하지 않는다. 행복 점수와 우울 점수가 모두 0인 경우($P_H = P_G = 0$)에 행복 지수는 0.5라고 가정한다.

예를 들어, 친구로부터 온 메시지가 다음과 같다고 하자.

"SAD MOVIES ALWAYS MAKE ME CRY"

A와 Y가 각각 4회, 2회 나타났으므로 행복 점수는 6이다 (A는 SAD에 한 번, ALWAYS에 두 번, MAKE에 한 번 나타나고, Y는 ALWAYS와 CRY에 각각 한 번씩 나타남). S와 A, D가 각각 3회, 4회, 1회 나타났으므로 우울 점수는 8이다 (S는 SAD와 MOVIES, ALWAYS 에 각 한 번, A는 행복 점수 산출 시와 마찬가지로 4번, D는 SAD에 한 번 나타남). $P_H = 6$이고 $P_G = 8$이므로 행복 지수는 $H = \frac{6}{14} = 0.4286$, 즉 42.86%로 산출된다. 행복 지수가 50%보다 낮으므로 친구는 우울하다.

짐의 친구의 문자 메시지가 주어졌을 때, 친구의 행복 지수를 계산하는 프로그램을 작성하라.

Input

입력은 표준 입력을 사용한다. 입력은 문자 메시지를 포함하는 한 행으로 구성된다. 메시지는 영문자 알파벳 단어로만 구성되는데, 단어는 공백으로 분리되어 주어진다. 각 단어는 대문자로만 구성된다. 단어의 최대 길이는 20이며 최대 단어 수는 80이다.

Output

출력은 표준 출력을 사용한다. 출력은 한 행으로 구성된다. 출력 행에는 산출된 행복 지수를 출력한다. 행복 지수는 백분율로 출력하되 백분율 기호(%)는 출력하지 않는다. 행복 지수는 소수점 이하 두 자리까지 반올림하여 출력한다. P_H와 P_G가 모두 0인 경우에 행복 지수는 0.5라는 점에 주의하자.

다음은 세 테스트 경우에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
`SAD MOVIES ALWAYS MAKE ME CRY`	`42.86`

Sample Input 2	Output for the Sample Input 2
`ICPC PROGRAMMING`	`75.00`

Sample Input 3	Output for the Sample Input 3
`WRITING`	`50.00`

Problem D
Palindrome Numbers
Time Limit: 1 Second

A positive integer P is called a palindrome number when the same numbers are produced by writing the digits in P in forwards and backwards. For instance, positive integers like 1, 101, and 12322321 are all palindrome numbers. Given a positive integer n, write a program that calculates the number of distinct palindrome numbers that are less than or equal to n. For example, when $n = 20$ there are 10 distinct palindrome numbers $1, 2, 3, 4, 5, 6, 7, 8, 9, 11$ that are less than or equal to 20.

Input
Your program is to read from standard input. The input starts with a line containing one positive integer n ($1 \leq n < 10^{10}$).

Output
Your program is to write to standard output. Print exactly one line. The line should contain the number of palindrome numbers that are less than or equal to n.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
`20`	`10`

Sample Input 2	Output for the Sample Input 2
`101`	`19`

Problem D
회문수
제한 시간: 1 초

어떤 양의 정수 P 에 대해 P 를 구성하는 숫자들을 왼쪽부터 적는 경우와 오른쪽부터 적은 결과가 서로 일치할 경우, P 를 회문수(palindrome number)라 한다. 예를 들어 $1, 101, 12322321$ 은 모두 회문수이다. 양의 정수 n이 주어졌을 때, n 이하의 서로 다른 회문수의 개수를 출력하는 프로그램을 작성하시오. 예를 들어 $n = 20$ 인 경우, 20 이하의 회문수는 총 10 개 $(1, 2, 3, 4, 5, 6, 7, 8, 9, 11)$ 존재한다.

Input
입력은 표준입력을 사용한다. 첫 번째 줄에 양의 정수 n $(1 \le n < 10^{10})$ 이 주어진다.

Output
출력은 표준출력을 사용한다. n 이하의 서로 다른 회문수의 개수를 한 줄에 출력한다.

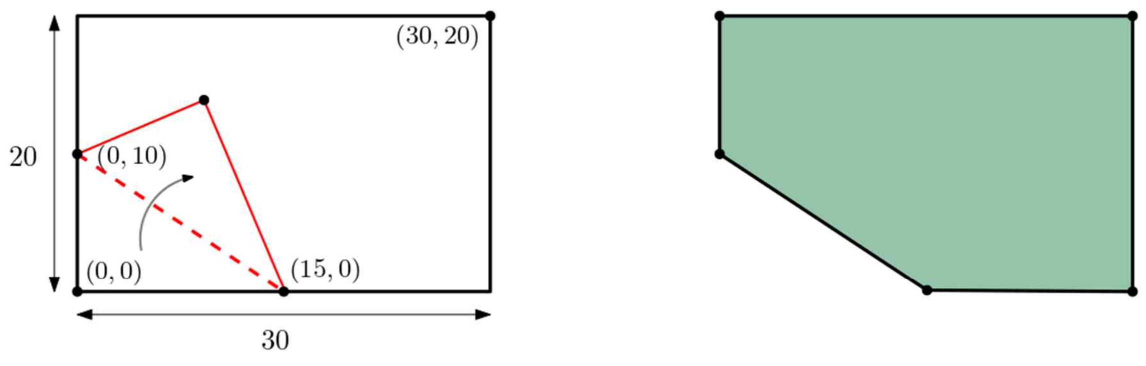
다음은 두 테스트경우에 대한 입출력 예이다.

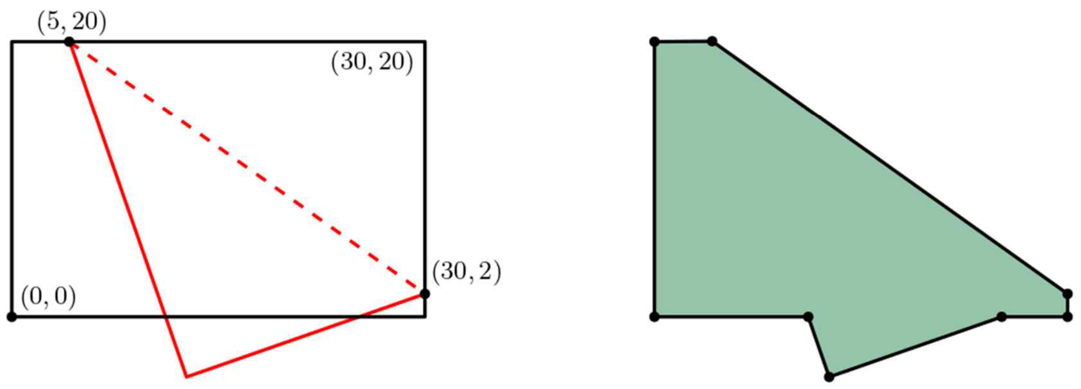
Sample Input 1	Output for the Sample Input 1
20	10

Sample Input 2	Output for the Sample Input 2
101	19

Problem E
Paper Folding
Time Limit: 1 Second

A rectangular paper is placed in the first quadrant with its vertex at the origin as shown in the figures below. We will fold this paper along a line which is specified by two points on the different edges of the paper. You are asked to compute the area of the 2D polygonal shape of the folded paper. In the following figures, you can see two papers and the corresponding fold lines (red dotted) and the folded paper (green polygon).





Input
Your program is to read from standard input. The first line contains two integers w and h, $10 \le w, h \le 1{,}000$ which represent the width and height of a rectangular paper, respectively. The coordinates of the two points that determine the folding line will be given at the second line and the third line. Note that two folding points are on the two different edges of the rectangle.

Output
Your program is to write to standard output. Print exactly one integer that is the integer part of the area of the folded polygon. For example, if the area of the folded polygon is 56.678, then you should print 56.

The following shows sample input and output for two test cases illustrated above.

Sample Input 1

```
30 20
0 10
15 0
```

Output for the Sample Input 1

```
525
```

Sample Input 2

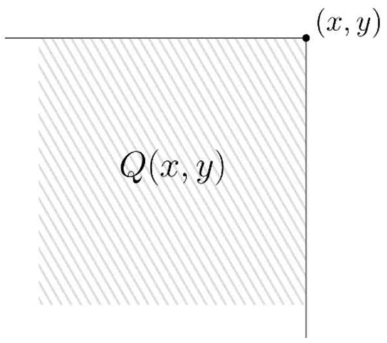
```
30 20
5 20
30 2
```

Output for the Sample Input 2

```
397
```

Problem F
Perfect Quadrants
Time Limit: 4 Seconds

Consider the plane and any point (x, y) in the plane. Now, you draw two half-lines starting from point (x, y), one downwards vertically and the other leftwards horizontally. The (infinite) region below the horizontal half-line and to the left of the vertical half-line is called a *quadrant*, denoted by $Q(x, y)$. Note that the two half-lines of $Q(x, y)$ form its boundary, while its interior excludes the boundary. See below.



Let L be a natural number and S be the set of points (x, y) in the plane with $x, y \in \{0,1,2, \dots, L\}$. So, S consists of $(L + 1)^2$ points. For some $k \geq 1$, you are given k finite subsets $P_1, P_2, \dots, P_k \subseteq S$ of S and k nonnegative integers $c_1, c_2, \dots, c_k$. We say that a quadrant Q is $(c_1, c_2, \dots, c_k)$-*perfect* when the following condition is satisfied for all $i = 1, 2, \dots, k$:

No points in P_i lie on the boundary of Q and $|P_i \cap Q| = c_i$.

Write a computer program that computes and prints out the number of points $(x, y) \in S$ such that $Q(x, y)$ is $(c_1, c_2, \dots, c_k)$-perfect.

Input
Your program is to read from standard input. The input starts with a line consisting of two integers, L and k ($1 \leq L \leq 10^9$, $1 \leq k \leq 100{,}000$). The second line of the input consists of k nonnegative integers $c_1, c_2, \dots, c_k$ ($0 \leq c_1, c_2, \dots, c_k \leq 1{,}000{,}000$). The third line consists of a single integer N ($1 \leq N \leq 1{,}000{,}000$), where N denotes the total number of input points, that is, $N = |P_1| + |P_2| + \cdots + |P_k|$. In each of the following N lines, three integers x, y, and i ($0 \leq x, y \leq L$, $1 \leq i \leq k$) are given, meaning that the point (x, y) is a member of the set P_i. You can assume that no axis-parallel line passes through two of the N input points.

Output
Your program is to write to standard output. Print exactly one line. The line should consist of a single integer, representing the number of points $(x, y) \in S$ such that $Q(x, y)$ is $(c_1, c_2, \dots, c_k)$-perfect.

The following shows sample input and output for three test cases.

Sample Input 1

```
10 1
1
9
0 3 1
8 0 1
4 1 1
6 2 1
7 7 1
2 5 1
3 6 1
1 8 1
5 4 1
```

Output for the Sample Input 1

```
7
```

Sample Input 2

```
10 1
2
9
0 3 1
8 0 1
4 1 1
6 2 1
7 7 1
2 5 1
3 6 1
1 8 1
5 4 1
```

Output for the Sample Input 2

```
0
```

Sample Input 3

```
10 2
1 1
8
1 4 1
2 7 2
4 6 1
5 3 2
6 5 1
7 2 2
8 1 1
9 9 2
```

Output for the Sample Input 3

```
3
```

Problem G
Reafy Sequence
Time Limit: 0.5 Seconds

It may sound weird, but Jaehoon has been obsessed with completely reduced fractions, that is, *irreducible fractions*. In particular, he is interested in listing all such fractions between 0 and 1 in increasing order. To take a more structured approach, Jaehoon defines $R(n)$ to be the sequence of ascending irreducible fractions with denominator less than or equal to n for an integer $n \geq 1$. He calls this sequence the *Reafy* sequence of order n. For example, the Reafy sequences of orders 1 to 5 are as follows:

$$R(1) = \left\{\frac{0}{1}, \frac{1}{1}\right\}$$

$$R(2) = \left\{\frac{0}{1}, \frac{1}{2}, \frac{1}{1}\right\}$$

$$R(3) = \left\{\frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1}\right\}$$

$$R(4) = \left\{\frac{0}{1}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{1}{1}\right\}$$

$$R(5) = \left\{\frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}\right\}$$

Given positive integers n and k, write a program to output the k-th fraction of $R(n)$. The first fraction of $R(n)$ is $\frac{0}{1}$ and the $|R(n)|$-th fraction is $\frac{1}{1}$.

Input

Your program is to read from standard input. The input is a line that contains two integers, n and k ($1 \leq n \leq 5{,}000$, $1 \leq k \leq |R(n)|$).

Output

Your program is to write to standard output. Print exactly one line. The line should contain two integers a and b, where the k-th fraction of $R(n)$ is $\frac{a}{b}$.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
`4 3`	`1 3`

Sample Input 2	Output for the Sample Input 2
`5 9`	`3 4`

Problem G
Reafy 수열
제한 시간: 0.5 초

이상하게 들릴지 모르지만, 재훈은 요즘 0과 1 사이의 기약 분수(irreducible fraction)를 오름차순으로 나열하는 것에 관심이 많다. 이를 위해, n차 수열 $R(n)$을 0과 1사이의 기약 분수 중에서 분모가 n 이하인 기약 분수의 오름차순 수열로 정의하고, 이를 *Reafy* 수열이라고 부르기로 했다. 여기서, n은 양의 정수이다.

예를 들어, 1차부터 5차까지의 Reafy 수열은 다음과 같다.

$$R(1) = \left\{\frac{0}{1}, \frac{1}{1}\right\}$$
$$R(2) = \left\{\frac{0}{1}, \frac{1}{2}, \frac{1}{1}\right\}$$
$$R(3) = \left\{\frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1}\right\}$$
$$R(4) = \left\{\frac{0}{1}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{1}{1}\right\}$$
$$R(5) = \left\{\frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}\right\}$$

두 양의 정수 n과 k가 입력으로 주어지면 Reafy 수열 $R(n)$의 k번째 기약 분수를 출력하는 프로그램을 작성하시오. $R(n)$의 첫 번째 기약 분수는 $\frac{0}{1}$이고 $|R(n)|$-번째 기약 분수는 $\frac{1}{1}$이다.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 두 양의 정수 n과 k가 주어진다. 두 정수의 범위는 $1 \leq n \leq 5{,}000$, $1 \leq k \leq |R(n)|$)이다.

Output

출력은 표준출력을 사용한다. $R(n)$의 k-번째 기약 분수가 $\frac{a}{b}$ 라면 a와 b 값을 차례대로 공백 하나를 사이에 두고 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	**Output for the Sample Input 1**
`4 3`	`1 3`

Sample Input 2	**Output for the Sample Input 2**
`5 9`	`3 4`

Problem H
Rigged Lottery
Time Limit: 2 Seconds

Recently, the ICPC Genie gave you the power to manipulate a single event! You remember that you bought a lottery ticket just yesterday and decide to use the power to manipulate the lottery results.

Every lottery ticket is a unique sequence of k numbers, and no two tickets have the same sequence. Each number in a lottery ticket is between 1 and c, and the same number can appear more than once in a ticket. For example, the following is the list of all possible lottery tickets when $k = 3$ and $c = 2$.

$$(1,1,1),\ (1,1,2),\ (1,2,1),\ (1,2,2),\ (2,1,1),\ (2,1,2),\ (2,2,1),\ (2,2,2)$$

The lottery results are determined as follows. First, a lottery machine randomly selects a finite sequence of numbers, where each number is between 1 and c and each number can occur more than once. This sequence is called the losing sequence. If the losing sequence has your ticket as a subsequence, then your ticket is a losing ticket. Otherwise, your ticket is declared as a winning ticket. Given a losing sequence, it is possible that there are several different winning tickets; in this case, the prize is split evenly.

For example, if the losing sequence is $(2,1,2,2,1)$, the lottery ticket of $(1,1,1)$ is a winning ticket, but the lottery ticket of $(2,2,2)$ is a losing ticket.

You decide to generate the losing sequence in which your lottery ticket is the only winning ticket. Since more winning tickets imply a smaller share of the prize, you want the losing sequence to have all sequences of length k as its subsequences except for your own lottery ticket.

Given numbers k and c, and the sequence of k numbers on your lottery ticket that you bought yesterday, write a program that outputs the shortest losing sequence in which your ticket is the only winning ticket. If there are two or more such losing sequences, then your program must output the lexicographically first sequence among them.

Input

Your program is to read from standard input. The input consists of three lines. The first line contains a positive integer k, where k is the length of a lottery ticket. The second line contains a positive integer c $(2 \le k + c \le 10{,}000)$, where c is the largest number in a lottery ticket. The third line contains k integers that represent your lottery ticket, where the integers are delimited by whitespace and each integer is between 1 and c.

Output

Your program is to write to standard output. Print exactly one line. The line should print the shortest sequence of numbers in which all sequences of length k, except for the given lottery ticket, appear as its subsequence. Each number should be separated by a whitespace. If there are two or more losing sequences, then print the lexicographically first sequence among them.

The following shows sample input and output for three test cases.

Sample Input 1

```
1
4
3
```

Output for the Sample Input 1

```
1 2 4
```

Sample Input 2

```
2
3
3 3
```

Output for the Sample Input 2

```
1 2 3 1 2
```

Sample Input 3

```
3
2
1 2 1
```

Output for the Sample Input 3

```
2 2 1 1 1 2 2
```

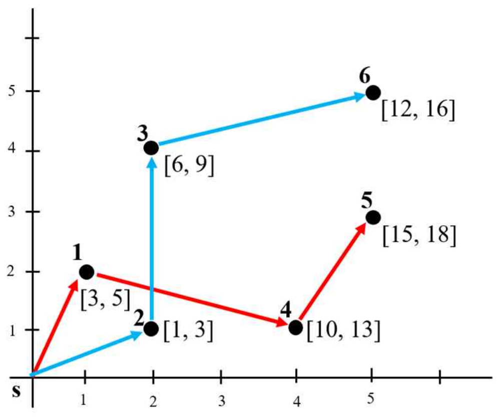
Problem I
Safari
Time Limit: 0.5 Seconds

Safari is a journey that involves going into nature to watch wild animals. Typically, safari participants travel through vast grasslands in four-wheel-drive cars, shortly 4WD cars. Imagine you are on safari in a 4WD car C.

Animals appear in specific places on the grassland, where you can consider the places as the points on the plane. There are n animals, indicated as 1 to n, in which the animal i appears at the point p_i in the plane. Each animal appears only in its own certain time interval. Specifically, the animal i appears in the time interval $[a_i, b_i]$. So, when the car C stays at p_i for the duration $[v, w]$, you have the opportunity to observe the animal i for the time period $[v, w] \cap [a_i, b_i]$. Note that if you observe an animal during $[\alpha, \beta]$, the length of time when you observe it is $|\beta - \alpha|$.

The car C departs from the starting point $s = (0, 0)$ at time 0 and it moves at speed 1. Thus time d has passed when C moves distance d. The distance is measured in the L^1-metric. That is, the distance $d(p_1, p_2)$ between points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is $|x_1 - x_2| + |y_1 - y_2|$. It always takes as long as the distance $d(p_1, p_2)$ while the car moves from p_1 to p_2. Also, the car may stay at a point as long as you need, if necessary. When your safari tour ends at the last sighting point of an animal, you want to know the longest possible time for which you observe the animals.

For example, the figure below shows six animals, indicated as 1 to 6, which appear at the coordinates $(1, 2)$, $(2, 1)$, $(2, 4)$, $(4, 1)$, $(5, 3)$, $(5, 5)$, respectively, of points in the plane. Let us also indicate as 1 to 6 the points of animals. The time intervals when the animals appear are also displayed. First, consider the case the car C is driving along the blue path. The car departs from s at time 0 and arrives at the point 2 at time 3. Departing from it immediately at time 3, the length of time when you observe the animal 2 is 0. Afterward, you arrive at the point 3 at time 6 and stay there during $[6, 8]$. Next, you arrive at the point 6 at time 12 and observe the animal 6 during $[12, 16]$. Then the total length of time when you observe the animals is $0 + 2 + 4 = 6$. Secondly, consider the case the car C is driving along the red path. At first, you arrive at the point 1 at time 3 and stay during $[3, 6]$. Departing from it at time 6, you arrive at the point 4 at time 10 and stay during $[10, 12]$. Then you arrive at the point 5 at time 15 and observe the animal 5 during $[15, 18]$. In this case, the total length of time when you observe the animals is $2 + 2 + 3 = 7$, which is longer than the blue path and actually, the length of the longest time when you can observe the animals.



Given n coordinates of points and n time intervals for the appearances of animals, write a program to output the length of the longest possible time when you observe the animals.

Input

Your program is to read from standard input. The input starts with a line containing one integer n ($1 \leq n \leq 5{,}000$), where n is the number of animals. The animals are numbered from 1 to n. In the following n lines, the i-th line contains two integers x_i and y_i that represent the coordinate (x_i, y_i) of the point p_i in the plane where the animal i appears ($0 \leq x_i, y_i \leq 10^6$). Note that the car C is located at $(0, 0)$ at time 0. The given points containing $(0, 0)$ are all distinct. In the following n lines, the i-th line contains two integers v_i and w_i that represent the duration $[v_i, w_i]$ when the animal i appears at p_i ($0 \leq v_i < w_i \leq 10^9$).

Output

Your program is to write to standard output. Print exactly one line. The line should contain the length of the longest time when you can observe the animals.

The following shows sample input and output for two test cases.

Sample Input 1

```
3
1 2
2 1
3 3
2 5
4 7
10 12
```

Output for the Sample Input 1

```
5
```

Sample Input 2

```
6
1 2
2 1
2 4
4 1
5 3
5 5
3 5
1 3
6 9
10 13
15 18
12 16
```

Output for the Sample Input 2

```
7
```

Problem J

Server Overload

Time Limit: 2 Seconds

The IT team of a SY company manages the sever log data. This log data is organized into an $n \times n$ grid, with each cell storing a number indicating the server access count during a specific time period. The server has recently been overloaded and is at risk of going down. To determine the cause of the server overload, the IT team uses a specialized 1×3 analysis tool to find the sub-grids of $n \times n$ grid that represent the high access counts. The 1×3 analysis tool covers some 1×3 sub-grid (of the $n \times n$ grid) of vertical length of 1 and of horizontal length of 3. The tool reports the sum of access counts stored in the cells of the sub-grid that the tool covers. The only limitation is that you can use this analysis tool at most k times and the 1×3 sub-grids covered by the tool should not overlap.

Given an $n \times n$ grid and a positive integer k, write a program that outputs the maximum of the total sum of access counts stored in cells of the $n \times n$ grid covered by the 1×3 analysis tool, such that the tool is used at most k times and no 1×3 sub-grids covered by the tool overlap.

Input

Your program is to read from standard input. The input starts with a line containing two integers, n and k ($3 \leq n \leq 1{,}000$, $1 \leq k \leq 5{,}000$), where n represents the size of the grid and k is the maximum number of times that the analysis tool can be used. In the following n lines, access count values of the $n \times n$ grid are given; the i-th line contains n access count values (from the first column to the last column) of the i-th row of the grid. All these access count values are integers between 1 and 1,000,000,000.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the maximum of the total sum of access counts in cells covered by the analysis tool such that the tool can be used at most k times and no 1×3 sub-grids covered by the tool overlap.

The following shows sample input and output for two test cases.

Sample Input 1	**Output for the Sample Input 1**

5 2
1 2 3 1 2
3 4 2 5 6
2 4 2 3 5
5 4 3 2 5
6 5 4 3 5
	28

Sample Input 2

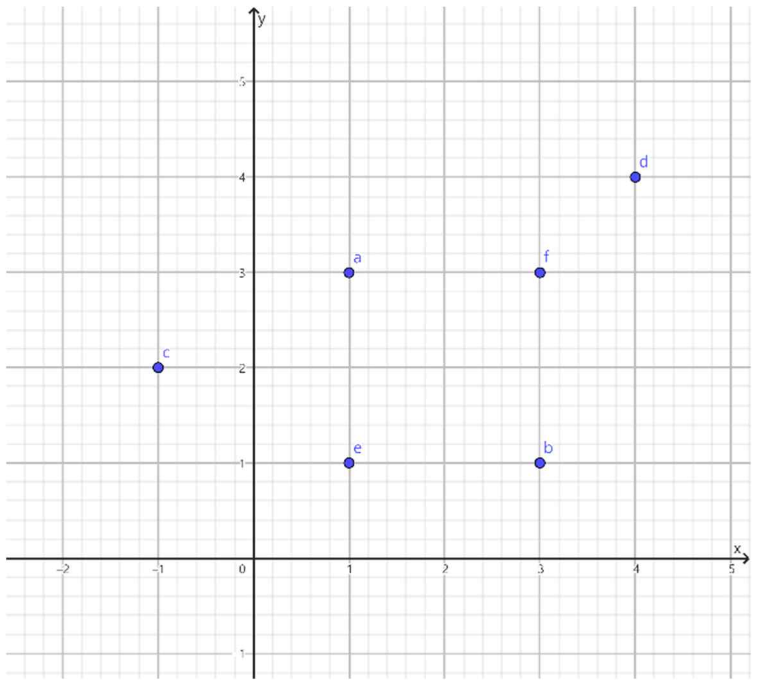
```
6 3
1 2 3 1 2 1
3 4 2 5 6 2
2 4 2 3 5 5
8 8 8 8 8 8
9 9 9 9 9 1
1 2 1 2 3 1
```

Output for the Sample Input 2

```
75
```

Problem K
Symmetry of Stars
Time Limit: 2 Seconds

Twinkling Stars in the universe attract us, guide us, and shed numerous intuitions to us. Astronomer Dr. K observed twinkling stars in a dark sky. One day, he was curious of symmetry of stars. To simplify the problem, he assumed the sky is a xy plane and the stars are points placed on the plane. When the set of stars S and a point $p = (p_x, p_y)$ are given, *symmetry of stars S with respect to a point p* is defined as the number of points $(x, y) \in S$ such that there exists at least one point $(x', y') \in S$ which satisfies $\left(\frac{x+x'}{2}, \frac{y+y'}{2}\right) = (p_x, p_y)$. When the set of stars S is given, *symmetry of stars S* is defined as the maximum symmetry of stars S with respect to any point p in the whole xy plane. Let's see an example following.



In the example above, we are given a set of stars $S = \{(1,3), (3,1), (-1,2), (4,4), (1,1), (3,3)\}$. The symmetry of stars S with respect to a point $p = (2,2)$ is 4 since the point $a = (1,3)$ has point $b = (3,1)$ which satisfies $\left(\frac{a_x+b_x}{2}, \frac{a_y+b_y}{2}\right) = \left(\frac{1+3}{2}, \frac{3+1}{2}\right) = (p_x, p_y) = (2,2)$ and the point $e = (1,1)$ has point $f = (3,3)$ which satisfies $\left(\frac{e_x+f_x}{2}, \frac{e_y+f_y}{2}\right) = \left(\frac{1+3}{2}, \frac{1+3}{2}\right) = (p_x, p_y) = (2,2)$. The symmetry of stars S with respect to a point $p = (-1,2)$ is 1 since the point $c = (-1,2)$ has point $c = (-1,2)$ itself which satisfies $\left(\frac{c_x+c_x}{2}, \frac{c_y+c_y}{2}\right) = \left(\frac{-1-1}{2}, \frac{2+2}{2}\right) = (p_x, p_y) = (-1,2)$. The symmetry of stars S is 4 since the symmetry of stars S with respect to the point $p = (2, 2)$ is the maximum among all the points in the xy plane.

Given a list of n distinct points that represent stars, write a program to output the symmetry of the given stars.

Input

Your program is to read from standard input. The input starts with a line containing one integer, n ($1 \leq n \leq 3{,}000$), where n is the number of stars. The stars are numbered from 1 to n. In the following n lines, the i-th line contains two integers that represent $x(-10^9 \leq x \leq 10^9)$ and $y(-10^9 \leq y \leq 10^9)$ coordinates of the star i, repectively. Note that no two stars are in the same position.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the symmetry of stars.

The following shows sample input and output for three test cases.

Sample Input 1

```
6
1 3
3 1
-1 2
4 4
1 1
3 3
```

Output for the Sample Input 1

```
4
```

Sample Input 2

```
5
1 3
3 1
1 1
3 3
2 2
```

Output for the Sample Input 2

```
5
```

Sample Input 3

```
1
1 5
```

Output for the Sample Input 3

```
1
```